Lassi Rasimus

ASP.NET-pohjaisen häiriötiedotussovelluksen toteutus

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tietotekniikka

Insinöörityö

29.4.2014

Metropolia

Tekijä(t) Otsikko Sivumäärä Aika	Lassi Rasimus ASP.NET-pohjaisen häiriötiedotussovelluksen toteutus 38 sivua 29.4.2014
Tutkinto	insinööri (AMK)
Koulutusohjelma	Tietotekniikka
Suuntautumisvaihtoehto	Ohjelmistotekniikka
Ohjaaja(t)	Järjestelmäasiantuntija Antti Kaleva Yliopettaja Erja Nikunen

Insinöörityön tavoitteena oli suunnitella ja toteuttaa tietoliikenneverkon häiriöistä tiedottamiseen soveltuva työkalu internetpalveluntarjoajien käyttöön. Teknisesti vaativa työ toteutettiin mahdollisimman pienillä kustannuksilla eli projekti pyrittiin rakentamaan ilmaisilla, avoimen lähdekoodin, tekniikoilla. Suurimpana kysymyksenä oli, löytyykö ilmaiskartastoista tarpeeksi kattava materiaali työn toteuttamiseen ja miten niiden jatkokehitys onnistuu, kun käytössä on vain vapaan lähdekoodin tekniikoita.

Sovellus rakentuu ASP.NET-ohjelmistokehyksen päälle, jonka serveripuolen koodi kirjoitettiin Visual Basic -ohjelmointikielellä. Sovellus sisältää kaksi osa-aluetta. Ylläpitäjän työkalulla käyttäjä hallitsee tiedotteita, jotka näytetään asiakkaalle karttanäkymänä. Karttanäkymä toteutettiin OpenLayers-JavaScript-kirjastolla, joka näyttää häiriöiden vaikutusalueet OpenStreetMap-kartalla.

Vapaan lähdekoodin tekniikat todettiin toimiviksi ja tähän projektiin hyvin soveltuviksi. Erityisesti OpenStreetMap-karttapalvelun sisällön laajuus teki vaikutuksen. Sovellukseen onnistuttiin toteuttamaan kaikki halutut toiminnallisuudet ja sen käyttöliittymästä saatiin sulava ja selkeä. Myös asiakas oli tyytyväinen lopputuotteeseen.

Avainsanat	ASP.NET, OpenLayers, OpenStreetMap

Author(s) Title Number of Pages Date	Lassi Rasimus Implementation of ASP.NET Based Fault Announcement System 38 pages 29 April 2014
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Specialisation option	Software Engineering
Instructor(s)	Antti Kaleva, System Specialist Erja Nikunen, Principal Lecturer

The purpose of the project described in this thesis was to design and implement an application for internet service providers to announce about communication network faults. The technically demanding work was implemented with as low cost as possible, so it was built with free open source technologies. The biggest question was whether free mapping services could offer sufficiently broad material for the implementation of the project and how further development would be accomplished, when using only open source technologies.

The application was built with ASP.NET framework, of which the server side code was written with the Visual Basic programming language. The application has two sections. With the administrator tool a user can manage announcements, which are shown to a customer as a map view. The map view was implemented with OpenLayers JavaScript library, which shows influence areas of faults in the OpenStreetMap map.

Open source technologies were found to be highly functional and well fitting for this project. Especially the scope of content of the OpenStreetMap service was impressive. All necessary functionality was implemented successfully and the user interface became smooth and clear. The customer was also satisfied with the end product.

Keywords	ASP.NET, OpenLayers, OpenStreetMap

Sisällys

Lyhenteet

ArcIMS	Arc Internet Map Server. Kartoituspalvelu.
ASP	Active Server Pages. Microsoftin kehittämä dynaamisten websivujen luomiseen tarkoitettu palvelinpuolen ohjelmointimenetelmä.
CLR	Common Language Runtimen. .NET Frameworkin virtuaalikone-komponentti.
GeoJSON	Geo JavaScript Object Notation. Formaatti maantieteellisten ominaisuuksien koodaukseen.
GML	Geographic Markup Language. Maantieteellisten ominaisuuksien merkkauskieli.
GPS	Global Positioning System. Satelliittipaikannusjärjestelmä.
GeoRSS	Geo Rich Site Summary. Standardi, jolla määritetään paikkatieto osaksi web-syötettä.
HTML	HyperText Markup Language. Kuvauskieli, jolla voidaan kuvata hyperlinkkejä sisältävää tekstiä.
HTTPS	Hypertext Transfer Protocol Secure. HTTP-protokollan ja SSL/TLS-protokollan yhdistelmä, jota käytetään tiedon suojattuun siirtoon internetissä.
IPTV	Internet Protocol Television. Internet-protokollan käyttöön perustuva teknologia niin televisio-ohjelman jakelussa kuin paluukanavassakin.
REST	Representational state transfer. Arkkitehtuurimalli ohjelmointirajapintojen toteuttamiseen.

SOAP	Simple Object Access Protocol. Tietoliikenneprotokolla, jonka pääasiallisena tehtävänä on mahdollistaa proseduurien etäkutsu.
VPN	Virtual Private Network. Virtuaalinen erillisverkko, jolla kaksi tai useampia verkkoja voidaan yhdistää julkisen verkon yli muodostaen näennäisesti yksityisen verkon.
WFS	Web Feature Service. Standardi, joka välittää vektoritietoja paikkatietoserveriltä.
WMS	Web Map Service. Standardi, jolla julkaistaan karttatietoja kuvina.
XML	Extensible Markup Language. Merkintäkieli.

1 Johdanto

Insinöörityössä toteutettiin teleoperaattorille monipuolinen ja helppokäyttöinen tapa ilmoittaa asiakkaille verkossa tapahtuvista huolloista ja häiriöistä. Viestintäviraston määräyksen mukaan:

> Teleyritysten on tiedotettava käyttäjille viestintäpalvelujen ja -verkkojen käyttöön vaikuttavista häiriöistä. Vakavissa tilanteissa yritysten on ilmoitettava häiriöstä myös Viestintävirastolle.
>
> Käyttäjille on kerrottava muun muassa:
>
> • palvelut, joihin häiriön vaikutukset kohdistuvat
> • häiriön alkamisajankohta
> • häiriön vaikutusalue
> • häiriön korjausaika-arvio.
>
> Häiriötiedotteet ja -kartat löytyvät teleyritysten verkkosivuilta.[12.]

Tämän pohjalta lähdettiin kehittämään karttapohjaista sovellusta, jolla operaattori voi määrittää häiriön tai huollon vaikutusalueen ja muut määräyksessä määritellyt tiedot.

Teknisesti vaativa työ toteutettiin mahdollisimman pienillä kustannuksilla eli sovellus pyrittiin rakentamaan ilmaisilla, vapaan lähdekoodin, tekniikoilla. Suurimpana kysymyksenä oli, löytyykö ilmaiskartastoista tarpeeksi kattava materiaali työn toteuttamiseen ja miten niiden jatkokehitys onnistuu, kun käytössä on vain vapaan lähdekoodin tekniikoita. Valittuihin tekniikoihin tutustutaan kattavasti tämän raportin alussa.

Työn alkuperäinen tilaaja on Savonlinnan seutukunnalla internet- ja TV-palveluja tarjoava Blue Lake Communications Oy, mutta sovellusta on sittemmin myyty jo useammalle verkkopalveluita tarjoavalle yritykselle. Kaikkia sovelluksen hankkineita yhdistää NETadmin-verkonhallintasovelluksen käyttö verkkojen hallinnassa, sillä tämä työ on toteutettu osaksi NETadmin-ympäristöä.

2 Sovelluksen määrittely

Sovellus päätettiin toteuttaa kahtena erillisenä osana. Tiedotuksen hallinta rakennettiin NETadmin 8.6 verkonhallintasovelluksen raporttisivuksi, koska haluttiin säästyä erillisen käyttäjien autentikoinnin toteutukselta, käyttää jo olemassa olevia palvelinratkaisuja sekä NETadminin tarjoamia kehitystyökaluja. Sovelluksen asiakasnäkymä toteutettiin erilliseksi web-sivuksi, joka upotetaan asiakkaan verkkosivuille. Tietoturvasyistä asiakasnäkymä ei voi sijaita samalla serverillä kuin NETadmin-asennus.

Kun uusi häiriötiedote lisätään noudatetaan kaavaa:

- Merkitään häiriöön liittyvät tiedot.
- Etsitään katuosoitteen perusteella tietty kohta kartalta.
- Merkitään kartalta alue.
- Tallennettaan häiriötiedote.

Sovelluksesta käyttäjän tulee pystyä muokkaamaan ja poistamaan häiriötiedotteita. Tiedotteista on myös tarpeen pitää historia-listaa, joka pitää myös olla ladattavissa Excel-tiedostona. Myös käyttäjien tekemistä muutoksista tehdään loki.

Koska NETadmin 8.6 on toteutettu ASP.NET 4.0 ohjelmistokehyksellä, se on valittava myös tässä projektissa käytettäväksi. Sovellus pyörii Windows Server 2008 palvelimella, jossa se käyttää MySQL 5.1 -tietokantaa. Kartan toteutukseen on valittu OpenStreetMap- sekä OpenLayers–tekniikat niiden ilmaisuudesta johtuen. Katuosoitteesta selvitetään karttakoordinaatit Maanmittauslaitoksen tarjoaman Maastotietokannan osoitteiden kyselypalvelulla.

3 OpenLayers-Javascript-kirjasto

Sovelluksen karttanäkymä toteutettiin OpenLayers-Javascript-kirjaston avulla. Sen avulla voidaan toteuttaa eri tarkoituksia varten erillisiä tasoja karttaan. Kirjaston tarjoamat palvelut käydään tarkemmin läpi tässä luvussa.

3.1 Yleistä

OpenLayers on vapaan lähdekoodin olio-ohjelmointia hyödyntävä JavaScript-kirjasto interaktiivisten web-karttojen luontiin. Selainpuolella toimiva OpenLayers ei vaadi mitään palvelinpuolen ohjelmistoja tai asetuksia, joten se on täysin ohjelmistokehysneutraali. OpenLayersin loi MetaCarta-niminen yksityinen yritys vuonna 2005, se julkaistiin vapaan lähdekoodin kirjastona vuonna 2006. Se on kasvanut pitkällekehitetyksi ja suosituksi ohjelmistokehykseksi, jolla on takanaan iso joukko kehittäjiä. [1, s. 8.]

OpenLayers oli vielä 2010-luvun alussa erittäin suosittu tekniikka karttasovellusten toteuttamiseen, mutta on sittemmin menettänyt suosiotaan ohjelmistopäivitysten puutteen vuoksi. Suomessa OpenLayersillä on toteutettu mm. Järvi-Suomen Energian sähkönjakelun keskeytyskartta, josta käyttäjä voi tarkastaa sähköjakelussa olevat häiriöt alueellisesti.

OpenLayers noudattaa Open Geospatial Consortium standardeja, joten se on yhteensopiva kaikkien merkittävien ja yleisimpien paikkatietoservereiden kanssa. Tämä mahdollistaa seuraavien palveluiden käytön:

- Web Map Service (WMS) on standardi, jolla julkaistaan karttatietoja kuvina. WMS-serveri palauttaa kartan bitmap-formaatissa kuten JPEG, PNG tai GIF.[13, s. 71.]
- Web Feature Service (WFS) standardi välittää vektoritietoja paikkatietoserveriltä ja sen avulla voidaan jakaa paikkatietoja standardi formaatissa [13, s. 71].

Tiedonvälitys voidaan hoitaa useilla rasteri- ja vektoritiedostomuodoilla kuten Geo Rich Site Summary (GeoRSS), Geo JavaScript Object Notation (GeoJSON) ja Geographic Markup Language (GML). OpenLayers on siis joustava paikkatietojen lukemisessa mutta myös tehokas resurssien säästäjä, sillä voidaan optimoida web-karttojen suorituskykyä luomalla strategioita siitä, miten paikkatietoja pyydetään ja haetaan selainpuolella.

Interaktiivisten web-karttojen luontiin OpenLayers tarjoaa valikoiman kontrolleja, jotka voidaan antaa loppukäyttäjän käyttöön. Kontrolleilla käyttäjä voi mm. panoroida, zoomata ja jopa lisätä ja muokata omaa sisältöään kartalle. [2.]

3.2 Karttojen rakenne

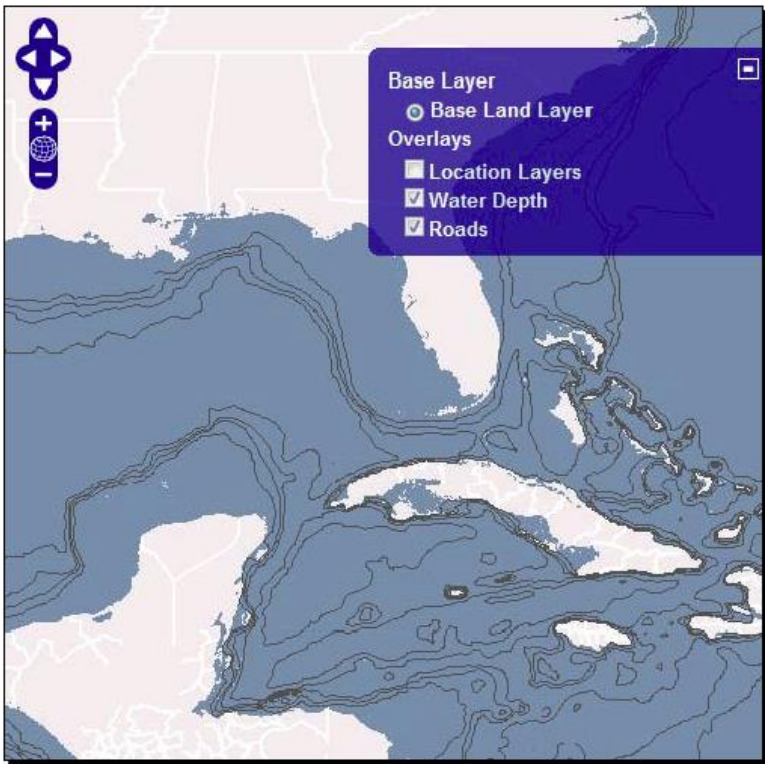
OpenLayers-kirjastoa käyttävät karttaohjelmistot koostuvat tasoista. Rakenne on verrannollinen paperiseen karttaan, jonka päälle asetellaan läpinäkyviä kalvoja, joihin on merkitty esimerkiksi kaupungin linja-autoreitit. Kalvojen poistaminen tai lisääminen vastaisi OpenLayers-kartassa tasojen piilottamista ja näyttämistä. Järjestyksellä, jolla kalvot asetetaan kartan päälle, on myös merkitystä. Jos eri kalvojen kuvioissa on päällekäisyyksiä, on näkyvillä aina ylempänä oleva kalvo. Näin on myös OpenLayers-kartassa. [1, s. 50.]

3.2.1 Pohjataso

Pohjataso eli Base Layer on aina tasolistan alimmaisena, eli kaikki muut tasot ovat sen päällä. Pohjataso on yleisesti karttataso. Ensimmäinen ohjelmaan lisätty taso toimii pohjatasona, mutta sen voi muuttaa myöhemmin. Pohjatasoja voi olla useampia, mutta vain yksi kerrallaan voi olla aktiivisena.

3.2.2 Yleistaso

Jokaista tasoa, joka ei ole pohjataso, kutsutaan yleistasoksi (Overlay Layer). Kuvan 1 tasojen vaihto kontrollissa pohjataso ja yleistasot on eroteltu toisistaan. Pohjatasona on yksinkertainen maarajat näyttävä kartta ja yleistasoina valittuna vedensyvyysrajat ja tiet. [1, s. 51.]



Kuva 1. OpenLayers-kartta, jossa näkyvillä tasojen vaihto kontrolli [1, s. 60].

3.3 Tasotyypit

OpenLayers tarjoaa useita eri tasotyyppejä. Kaikki tasot periytyvät OpenLayers.Layer-luokasta. Tässä luvussa esitellään niistä yleisimmät.

ArcGIS93Rest-taso

ArcGIS93Rest-taso mahdollistaa vuorovaikutuksen ArcGIS Server 9.3:n kanssa, jota käytetään paikkatietojärjestelmien Webpalveluiden, ohjelmistojen ja tietojen luomiseen ja ylläpitoon. Yhteydenpito hoidetaan palvelun tarjoaman Representational state transfer (REST) rajapinnan kautta. [1, s. 72.]

ArcIMS-taso

ArcIMS-taso mahdollistaa tiedon näyttämisen Arc Internet Map Server (ArcIMS) kartoituspalvelusta. ArcIMS on paikkatietojärjestelmä, joka on suunniteltu karttojen toimittamiseen internetissä. [1, s. 73.]

Bing-taso

Bing-tasolla ohjelman käyttöön saadaan Microsoftin Bing -karttapalvelut Bing Maps REST rajapinnalla. Käytettävissä olevia Bingin karttatyyppejä ovat katunäkymä, tienäkymä ja ilmakuva. [5; 6.]

Google-taso

Google-tasolla ohjelman käyttöön saadaan Google Maps-palvelut Google Maps -palvelut Googlen tarjoamalla ohjelmointirajapinnalla. Erilaisia karttavaihtoehtoja on neljä:

- ROADMAP näyttää perustiekartan. Tämä on vakiokarttatyyppi.
- SATELLITE näyttää Google Earth satelliittikuvia.
- HYBRID näyttää sekoituksen normaaleja ja satelliittikuvia.
- TERRAIN näyttää fyysisen kartan, joka perustuu maastotietoihin. [1, s. 73-74; 3.]

Ruudukkotaso

Grid on pohjataso, josta useat muut tasot periytyvät. Se rakentaa ruudukon ja liittää siihen ruutuja eli karttakuvia. Grid-taso käyttää HTTPRequest-luokkaa kommunikointiin karttaserverin kanssa saadakseen karttakuvat, joita se tarvitsee kartan rakentamiseen. Grid-tasosta periytyvät tasot toimivat samalla periaatteella. [1, s. 75.]

Kuvataso

Image-taso mahdollistaa kuvan käyttämisen karttatasona. Se toimii hieman eri tavalla kuin muut OpenLayers-tasoluokat. Taso lähettää vain yhden pyynnön saadakseen kuvan käyttöönsä. Tämän jälkeen selaimella on kuva käytössään, OpenLayers hoitaa kaiken vuorovaikutuksen sen kanssa. Mitään muita pyyntöjä ei lähetetä serverille ensimmäisen pyynnön jälkeen. Image-tasossa ei ole ruudukkoja, kuten muissa Grid-tasosta periytyvissä karttatasoissa. Kuvassa 2 kuvatasoa käytetään pohjatasona, jolloin siihen voidaan merkitä tietoja yleistasoina. Kuvatasoa käytetään pohjatasona lähinnä silloin kun halutaan luoda omatekoisia karttoja esimerkiksi rakennuksista tai pienistä alueista. [1, s. 76.]

Kuva 2. OpenLayers-kartta, jossa käytössä Image-taso pohjatasona [1, s. 77].

MapGuide-taso

MapGuide-tasolla voidaan ottaa käyttöön vapaan lähdekoodin karttojen luontityökalu MapGuide Open Source. MapGuide on nopea työkalu web-pohjaisten kartoitusohjelmien ja geomaattisien palveluiden käyttöön ja toteutukseen. [1, s. 78; 4.]

OMS-taso

OMS-tasolla ohjelman käyttöön saadaan OpenStreetMap-karttapalvelut. OpenStreetMap on avoin yhteistyöprojekti vapaasti muokattavien karttojen luomiseksi. Lisää aiheesta luvussa kolme. [5.]

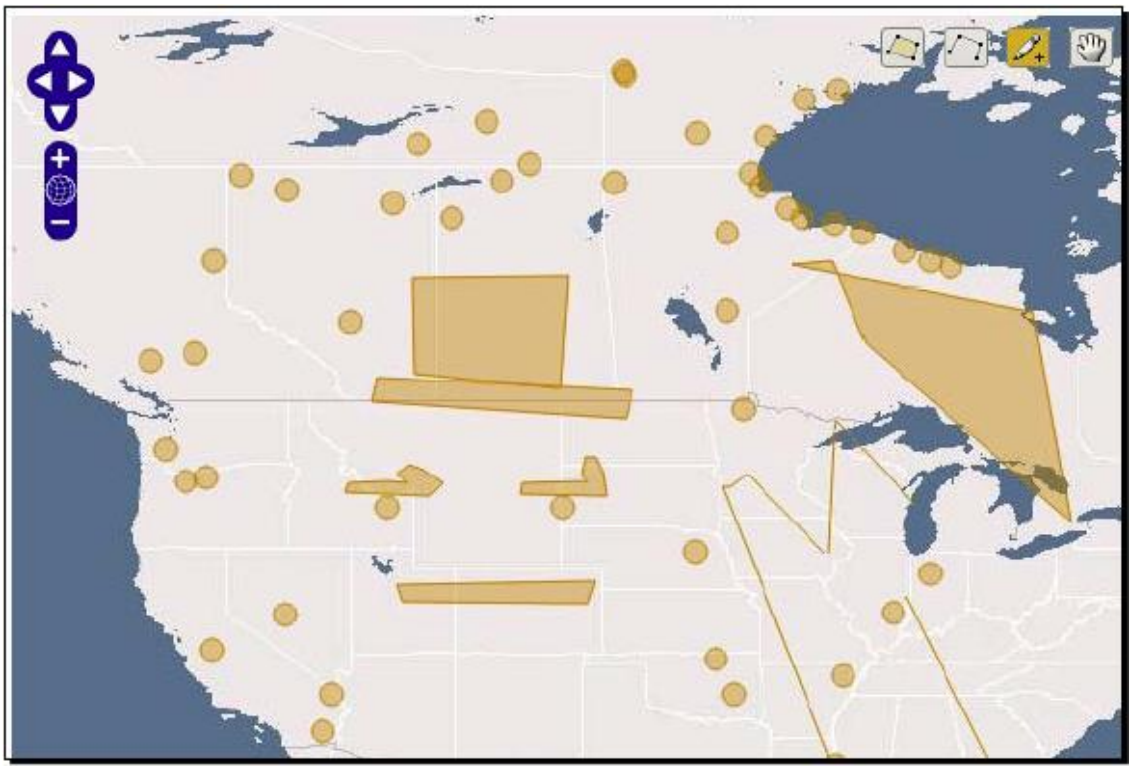
TileCache-taso

TileCache-tasolla otetaan käyttöön TileCatche, vapaan lähdekoodin ohjelmisto, joka vie serverin välimuistiin WMS-pyyntöjä suorituskyvyn parantamiseksi. Tämä tarkoittaa että heti kun pyyntö on lähetetty, vastaus eli karttakuva tallennetaan serverin

kovalevylle tai muistiin, josta se välittömästi palautetaan selaimelle. Jatkossa kuva on suoraan ohjelmiston käytettävissä ilman uutta WMS-pyyntöä. [1, s. 79.]

Vektoritaso

Vector-taso eli vektoritaso on yksi OpenLayersin hyödyllisimmistä tasotyypeistä. Sen avulla voidaan piirtää pisteitä ja polygoneja kartalle ja muotoilla niitä vapaasti. Myös interaktiivista toiminnallisuutta voidaan lisätä vektoritasoon, kuten objektia klikatessa avautuva ponnahdusikkuna tai käyttäjälle mahdollisuus muokata ja lisätä vektorikuvioita. Vektorien lisäykseen OpenLayers tarjoaa valmiin kontrollin, jolla käyttäjä voi kartalta pisteitä valitsemalla muodostaa alueita tai lisätä valmiita kuvioita. Kuvassa 3 on lisätty kartalle sekä ympyräkuvioita että alueita. Vektoritason toiminnallisuudet tapahtuvat kokonaisuudessa selainpuolella, joka mahdollistaa nopean ohjelman suorituksen. Vektoritasoon tehdyt muutokset voidaan kuitenkin viedä serverille tarpeen tullen. Näin käyttäjän tekemät muutokset voidaan ottaa käyttöön myös serveripuolen koodissa. [1, s. 228-229.]



Kuva 3. OpenLayers-kartta, jossa käytössä vektoritaso ja vektorien piirto -kontrolli [1, s. 231].

4 OpenStreetMap-karttaprojekti

Sovellukseen tarvittiin karttapalvelu, joka tarjoaisi riittävän hyvän karttatietokannan. OpenStreetMap-kartaston tietokannan rakennetta ja karttaprojektin toimintaa selitetään tässä luvussa.

4.1 Yleistä

Tässä projektissa toteutettava sovellus käyttää OpenStreetMap-kartastoa häiriötiedotteiden vaikutusalueiden esittämiseen. OpenStreetMap on projekti, jonka tarkoituksena on luoda ilmainen maantieteellinen tietokanta koko maailmasta niin, että lopulta tietokanta kattaisi kaikki maailman maantieteelliset ominaisuudet. Projekti alkoi teiden kartoittamisella, mutta se on kasvanut jo kattamaan polut, rakennukset, vesireitit, putkistot, metsät, rannat, postilaatikot ja jopa yksittäiset puut. Fyysisten elementtien lisäksi OpenStreetMap kattaa myös hallinnollisia rajoja, maankäytön yksityiskohtia, linja-autoreittejä ja muita abstrakteja ideoita, jotka eivät suoraan selviä maisemakuvista.

Tietokantaa kasvattavat kartoittajat, jotka keräävät tietoa liikkumalla kävellen tai ajaen ympäristössään. Heidän liikkeensä tallennetaan Global Positioning System (GPS) satelliittipaikannusjärjestelmän avulla. Kartoittajat kantavat GPS-vastaanottimia, joiden tiedoista voidaan luoda pisteitä ja linjoja, joista luodaan karttoja. Dataa voidaan kerätä myös olemassa olevista kartoista, jos niitä ei suojaa tekijänoikeudet. Tämä vaatii hieman lisätyötä tietojen päivittämisessä ja siistimisessä.

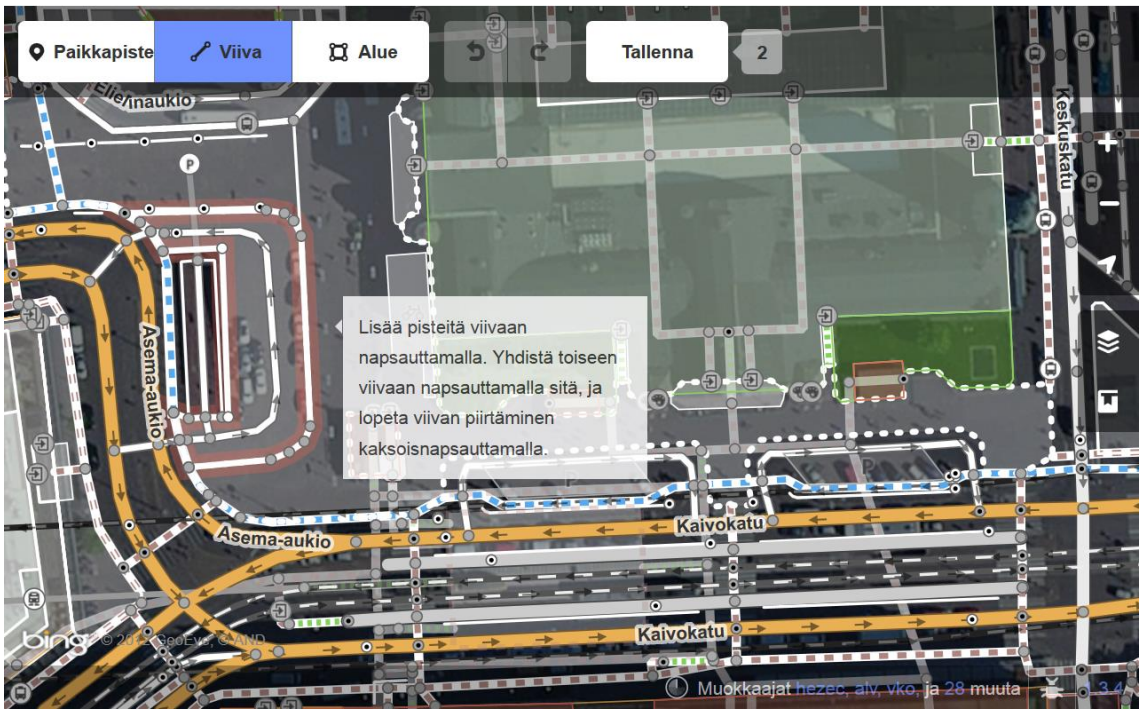
OpenStreetMap-tietokanta käyttää wiki-tyyppistä systeemiä, jossa kuka tahansa voi lisätä tai muokata minkä tahansa alueen tietoja. Muokkaustiedoista pidetään kattavat varmuuskopiot, jotta alkuperäiset tiedot voidaan palauttaa, ne osoittautuvat vääriksi. OpenStreetMap käyttää täysin omaa ohjelmistoa ja tietomallia, jotta näin suuren datamäärän käsittely olisi mahdollisimman helppoa ja joustavaa.

Suurin osa kartoittajista on vapaaehtoisia, jotka työskentelevät OpenStreetMapin parissa vapaa-aikanaan, mutta myös yritykset, järjestöt ja jopa hallitukset ovat edistäneet projektia. Vapaaehtoisten ihmisjoukkojen käyttöä tämänkaltaisissa projekteissa kutsutaan joukkoistamiseksi. Kyseessä on uusi ilmiö, joka perustuu

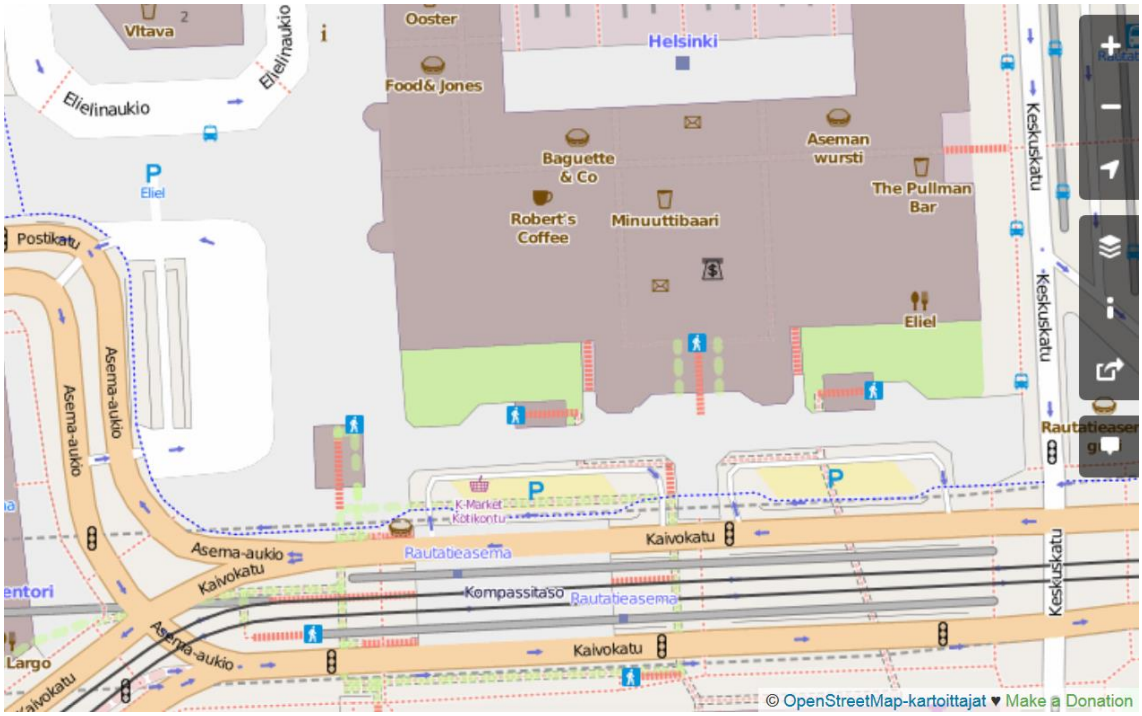
internetin hyödyntämiseen tehtävien jaossa ja tulosten keräyksessä. Laajakaistayhteyksien yleistyttyä siitä on tullut suosittu ja tehokas tapa kerätä suuret määrät tietoa pienellä rahoituksella. [7, s. 8-11.]

4.2 Rakenne

OpenStreetMap käyttää omaa datamalliaan, joka koostuu kolmesta yksinkertaisesta datatyypistä: solmuista (nodes), reiteistä (ways) ja yhteyksistä (relations). Kuvassa 4 on OpenStreetMap.org-sivuston tarjoama kartan muokkaus käyttöliittymä, josta näkyy hyvin esimerkiksi teiden muodostuminen solmuista. Kuvasta 5 näkyy, miltä sama alue näyttää normaalissa kartan tarkastelussa. Lisäksi tarjotaan vapaamuotoinen tietojen merkintärakenne, jonka avulla voidaan kuvailla tarkasti lähestulkoon mikä tahansa maantieteellinen objekti. Datamalli myös kuvaa objektien topologiaa eli miten ne ovat yhteydessä toisiinsa ja miten liikkua niiden välillä. Tämä on tärkeää varsinkin navigointiohjelmistoille.



Kuva 4. OpenStreetMapin muokkausnäkymä Helsingin rautatieasemasta.



Kuva 5. OpenStreetMap-kartta Helsingin rautatieasemasta.

4.3 Tietotyypit

Matemaattisin termein selitettynä OpenStreetMapin datamalli koostuu kulmista tai kärkipisteistä ja reunoista. Tälläisen verkon osat voivat olla yhdistettynä toisiinsa tai erillään. Vakioformaatti datamallin esittämiselle on Extensible Markup Language (XML) merkintäkieli. Se on myös ainoa formaatti, jossa dataa voi ladata OpenStreetMap-serveriltä. [7, s. 53-55.]

4.3.1 Solmu

Solmut ovat pisteitä tilassa. Ne ovat ainoita datatyyppejä, joilla on sijaintiedot, kaikki muut datatyypit turvautuvat solmuihin sijainnin selvittämisessä. Solmuja voidaan käyttää merkitsemään kiinnostavaa sijaintia, risteystä reittien välillä tai vain reitin suunnan muutosta.

```
<osm version="0.6" generator="OpenStreetMap server">
```

```
   <node id="483034256" lat="55.9458449" lon="-3.2035477" ver
       sion="1" changeset="2369219" user="spytfyre" uid="166957"
       visible="true" timestamp="2009-09-04T13:35:42Z">
      <tag k="name" v="The Blue Blazer"/>
      <tag k="amenity" v="pub"/>
   </node>
</osm>
```

Koodiesimerkki 1. Solmun XML-merkintä [7, s. 56].

Jokaisen solmun leveys- ja pituuskoordinaatti merkintään maksimissaan seitsemän desimaalin tarkkuudella (koodiesimerkki 1), joka antaa sijainnin tarkkuudeksi leveysvyöhykkeellä noin 1 cm ja pituusvyöhykkeellä noin 0,6-1 cm. Sijainnin lisäksi solmusta merkitään mm. sen ID, versionumero, viimeksi muokannut käyttäjä, ja aikaleima, jos kyseessä on sijaintipiste myös pisteen nimi ja tyyppi merkitään. [7, s. 55-57.]

4.3.2 Reitti

Reitit ovat järjestettyjä solmulistoja. Ne voivat kuvailla lineaarisia objekteja kuten teitä, polkuja ja vesireittejä, mutta ne voivat olla myös suljettuja, jolloin ne muodostavat alueita. Vaikka reitti koostuu solmuista, reitin tiedot eivät muutu vaikka solmuja muokattaisiin eikä solmun tiedoista selviä, onko se osa reittiä. Tiellä täytyy olla vähintään kaksi ja enintään 2000 solmua. Yläraja on asetettu, jotta openstreetmap.org-palvelimet eivät rasitu liikaa liian pitkistä reiteistä. Solmu taas voi kuulua rajoittamattomaan määrään reittejä.

Reiteillä katsotaan aina olevan suunta, vaikka se ei aina olekaan merkittävä tieto. Jos reitin ensimmäinen ja viimeinen solmu ovat samoja, reitistä muodostuu suljettu alue. Jos tarvitaan aluetta, joka on monimuotoisempi kuin yksinkertainen polygoni, voidaan käyttää useampaa reittiä ja yhdistää ne yhteysdatatyypillä.

```
<osm version="0.6" generator="OpenStreetMap server">
   <way id="43157302" visible="true" timestamp="2009-10-
       26T10:45:09Z" version="1" changeset="2954960" user="Ed
       Avis" uid="31257">
       <nd ref="540653724"/>
       <nd ref="25507043"/>
       <nd ref="107762"/>
       <nd ref="25507038"/>
       <nd ref="107759"/>
```

```
      <tag k="highway" v="primary"/>
      <tag k="lcn_ref" v="6a"/>
      <tag k="name" v="Parliament Street"/>
   </way>
</osm>
```

Koodiesimerkki 2. Reitin XML-merkintä [7, s. 58].

Reitin XML-merkintä koostuu perusmuokkaustietojen lisäksi listasta solmuja ja tagi-merkintöjä. nd-elementti merkitsee solmua, jonka ID-arvo merkataan atribuutilla ref (koodiesimerkki 2). Tagi-merkinnät antavat lisätietoja reitistä. Reitin kuvaus ei kerro, mihin muihin reitteihin se on yhteydessä. Tämä selviää ainoastaan siitä, mihin muihin reitteihin kyseisen reitin solmut kuuluvat. [7, s. 57-59.]

4.3.3 Yhteys

Yhteydet ovat listoja solmuista, reiteistä ja myös yhteyksistä. Yhteyksiä tarvitaan, jotta voidaan merkitä objekteja, joita ei voida kuvata vain yhdellä solmulla tai reitillä tai jos useampi saman tyypin objekti on päällekäin. Yhteyksiä tarvitaan esimerkiksi monimutkaisien haarautuvien katujen, pitkien teiden ja risteyksien kääntymisrajoituksien merkitsemiseen.

```
<osm version="0.6" generator="OpenStreetMap server">
   <relation id="113421" visible="true" timestamp= "2009-11-
      03T10:08:27Z" version="2" changeset="3023369" us-
      er="Jonathan Bennett" uid="5352">
      <member type="node" ref="270186" role="via"/>
      <member type="way" ref="4418767" role="from"/>
      <member type="way" ref="4641665" role="to"/>
      <tag k="restriction" v="no_right_turn"/>
      <tag k="type" v="restriction"/>
   </relation>
</osm>
```

Koodiesimerkki 3. Yhteyden XML-merkintä [7, s. 59].

Koodiesimerkki 3 havainnollistaa, miten merkitään risteyksen kääntymisrajoitus. Member-elementti ilmaisee yhteyteen kuuluvat osat. Elementistä kerrotaan sen tyyppi, ID ja rooli yhteydessä. Esimerkissä kerrotaan oikeallekääntymiskiellosta kahden tien risteyksessä. Kyseistä yhteyttä ei näytetä OpenStreetMap-kartalla, vaan sitä käyttävät reititysalgoritmit. [7, s. 59.]

5 ASP.NET-ohjelmistokehys

Tämä projekti on toteutettu ASP.NET-ohjelmistokehyksellä. Se valittiin, koska sovellus tehdään osaksi sitä käyttävää NETadmin-ohjelmistoa. 2000-luvun alussa Microsoft aloitti Active Server Pages (ASP) –teknologian seuraajan kehityksen. Tammikuussa 2002 julkaistiin ASP.NET 1.0 yhdessä .NET Frameworkin version 1.0 kanssa. ASP.NET on web-ohjelmistokehys, jota Microsoft kehittää ja markkinoi. Sen avulla ohjelmoijat voivat rakentaa dynaamisia web-sivuja, web-ohjelmia sekä web-palveluja. ASP.NET rakennettiin Common Language Runtimen (CLR) päälle, jotta ohjelmoijat voivat kirjoittaa ASP.NET-koodia millä tahansa tuetulla .NET-kielellä. ASP.NET Simple Object Access Protocol (SOAP) -lisäkehys mahdollistaa SOAP-viestien prosessoimisen ASP.NET-komponenttien toimesta.

5.1 Yleistä

ASP.NET web-sivut ovat ASP.NET-ohjelmistokehityksen rakennuspalikoita. Sivut sisältyvät .aspx-tiedostopäätteisiin tiedostoihin. Näissä tiedostoissa käytetään yleensä HyperText Markup Language (HTML) merkintäkieltä, jonka seassa on serveripuolen Web-kontrolleja. Myös serveripuolella suoritettavaa dynaamista koodia voidaan upottaa sivulle. ASP.NET Framework 2.0 esitteli code-behind-mallin, jossa staattinen teksti jää .aspx-sivulle, mutta dynaaminen koodi siirtyy .aspx.vb-, .aspx.cs- tai .aspx.fs –kooditiedostoon riippuen valitusta ohjelmointikielestä. [8.]

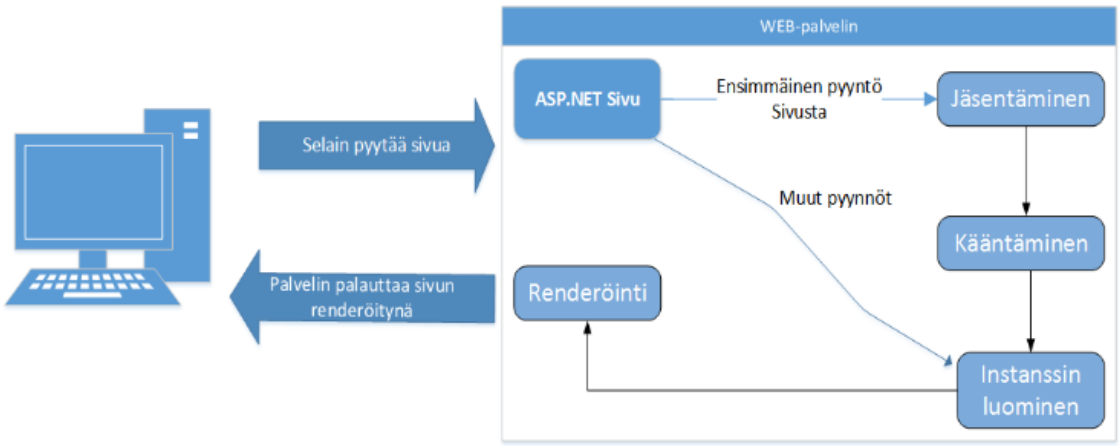
5.2 ASP.NET-sivustonkehitysmallit

ASP.NET tukee kolmea erilaista web-sivuston kehitysmallia: Web Pages, Model View Controller (MVC) ja Web Forms. Web Pages on kehitysmalleista yksinkertaisin, se perustuu klassiseen ASP-malliin, jossa kaikki koodi kirjoitetaan samaan tiedostoon. MVC-malli jakaa web-ohjelmistot kolmeksi eri komponentiksi: Malli (Model) kuvaa järjestelmän tiedon tallentamisen, ylläpidon ja käsittelyn, Näkymä (View) määrittää käyttöliittymän ulkoasun ja tietojen esityksen käyttöliittymässä ja Käsittelijä (Controller) vastaanottaa käyttäjältä tulevat käskyt sekä muuttaa mallia ja näkymää vastauksena niihin. Web Forms on perinteinen ASP.NET-kehitysmalli, joka perustuu tapahtumakeskeisiin sivuihin ja code-behind-malliin.[9.]

Tämä projekti on toteutettu ASP.NET Frameworkin 4.0-versiolla, Web Forms kehitysmallilla sekä Visual Basic ohjelmointikielellä, joten jatkossa keskitymme niiden käsittelyyn.

5.3 ASP.NET-sivun kääntäminen

ASP.NET-sivu käännetään palvelimella, kun selain pyytää sivua. Tällöin kuvan 6 mukaisesti sivu jäsennetään, käännetään ja luodaan sivusta muistinsisäinen instanssi. Instanssi renderöidään selaimen ymmärtämään HTML-muotoon ja palautetaan pyytäjälle. Lopuksi instanssi tuhotaan, joten instanssin käyttämä data ei ole enää käytössä seuraavalla sivun latauskerralla. Datan pysyvyys hoidetaan sessioilla ja kontrolleilla. [10, s. 3-9; 8.]



Kuva 6. ASP.NET-sivun kääntäminen [10, s. 6].

5.4 Palvelinkontrollit

ASP.NET-sivun käyttöliittymä rakentuu HTML-koodin lisäksi serverikontrolleista. Jokaisesta kontrollista muodostuu oma olioinstanssi, jota voidaan hallita kooditiedostossa. Kooditiedoston avulla saadaan .NET-sovelluskehyksen ohjelmointiominaisuudet käyttöön sivulla. Kontrollit ”elävät” serveripuolella ASPX-sivun sisällä, johon kääntäjä pääsee käsiksi suorituksen aikana.

Kontrollit määritellään seuraavan kaavan mukaisesti:

<asp:TypeOfControl ID="ControlName" runat="server" />

Näiden pakollisten perustietojen lisäksi kontrollille voi vielä määritellä kontrollikohtaisia atribuutteja. Kontrollin ID on aina oltava uniikki kyseisellä sivulla. Runat-attribuutin server-arvo kertoo, että kontrolli kuuluu prosessoida serveripuolella.

Osa kontrolleista kuvastavat suoraviivaisesti tiettyä HTML-elementtiä. Toiset taas ovat monimutkaisia abstrakteja elementtejä, jotka kuvastavat usean HTML-elementin kokonaisuutta.

```
<input type="text" value="Hello World" />
<asp:TextBox ID="Message" runat="server" Text="Hello World" />
```

Koodiesimerkki 4. Tekstilaatikon HTML- ja ASP.NET-toteutukset.

Koodiesimerkki 4 havainnollistaa tekstilaatikon HTML- ja ASP.NET –toteutuksien yhteneväisyyttä. Sivun renderöinnissä alempana olevasta ASP.NET-toteutuksesta muodostuu ylemmän muotoinen HTML-elementti. Koska ASP.NET-kontrollit prosessoidaan serveripuolella, niiden arvoja voidaan vapaasti muokata koodisivulla. Tässä tapauksessa esimerkiksi laatikon sisällä olevaa tekstiä voitaisiin muokata Visual Basic kielen komennolla `Message.Text = ”Muutettu teksti”.`

```
<asp:Repeater ID="repeaterEsim" runat="server" Visible="true">
    <HeaderTemplate>
        <table style="min-width: 50%;">
            <tr>
                <td><b>Kesto</b></td><td><b>Tyyppi</b></td>
            </tr>
    </HeaderTemplate>
    <ItemTemplate>
         <tr>
             <td><%#Container.DataItem("kesto")%></td>
             <td><%#Container.DataItem("tyyppi")%></td>
         </tr>
    </ItemTemplate>
    <FooterTemplate>
        </table>
    </FooterTemplate>
</asp:Repeater>
```

Koodiesimerkki 5. Repeater-kontrollin toteutus.

Esimerkkinä monimutkaisista elementeistä koodiesimerkki 5:n Repeater-kontrolli renderöityy kaksi sarakkeiseksi HTML-taulukoksi. Taulukon rivien määrä riippuu kontrollille annettavan datan määrästä. `Container.DataItem` määrittää, mitä dataa tulostetaan mihinkin kohtaa taulukkoa. Repeaterin datan lähde annetaan koodisivulla käyttäen sen DataSource-attribuuttia ja lopuksi sitoutetaan data kontrolliin komennolla DataBind(). ASP.NET sisältää laajan valikoiman erilaisia kontrolleja, jotka helpottavat ohjelmistokehittäjän työtä huomattavasti. Ohjelmistokehyksen mukana tulevien kontrollien lisäksi niitä on myös mahdollista luoda itse tai ottaa käyttöön yhteisön luomia kontrolleja. [11, s. 107-112.]

Kontrolleja voidaan myös luoda dynaamisesti koodisivulta. Tällöin niiden käsittely vastaa täysin normaalia olio-ohjelmointia. Dynaaminen kontrollien luonti tapahtuu yleisesti lisäämällä aspx-sivulle paneeleita. Paneelit vastaavat HTML-merkkauksessa div-elementtiä eli ne ovat alueita sivulla, joiden sisälle elementtejä voidaan lisätä. Kun haluttu kontrolli on luotu koodisivulla, se voidaan lisätä paneeliin. Renderöidystä HTML-koodista dynaamisesti luodut kontrollit erottaa yleensä ID-attribuutista, sillä dynaamisesti luoduille kontrolleille sitä ei yleensä erikseen lisätä, vaan kääntäjä luo sen renderöinnin aikana.

Dynaamisia kontrolleja käytetään useimmiten, kun sivulle on tarve rakentaa käyttöliittymä käyttäjän antaman tai liittyvän datan mukaisesti. Tällöin dynaamiset kontrollit ovat edukseen, muuten niiden käsittelystä tulee sivun koon kasvaessa monimutkaista. ASP.NET-sivu on mahdollista luoda lähes kokonaan dynaamisesti, kontrollien lisäksi sivulle voidaan antaa HTML-koodia ja muotoiluja koodisivulta. Tätäkään ei suositella, sillä dynaamisesti luotaessa kehittäjän on vaikeaa hahmottaa sivun ulkomuotoa.

5.5 ASP.NET-tilamoottori

Postback-tapahtumassa (sivun takaisinlähetys) muilla web-kehitystekniikoilla täytyy yleisesti lisätä alkuperäisen sivun lisäksi toinen sivu, joka käsittelee käyttäjän antamat arvot. Sivun käyttöliittymäelementtejen arvot myös katoavat jos niitä ei erikseen aseteta takaisin sivulle POST-arvoista. ASP.NET tilamoottori (state engine) poistaa tämän tarpeen.

ASP.NET Web Form lähettää sivun aina takaisin itselleen. Kontrollien arvot pysyvät tallella View State toiminnallisuuden ansiosta. Renderöinnissä HTML-koodiin lisätään automaattisesti koodiesimerkki 6:n mukainen piilotettu tekstikenttä. Kentän arvo kertoo tietoja sivusta, jotka lähetetään takaisin serverille postback-tapahtuman yhteydessä. Tämän jälkeen ASP.NET luo uuden sivun serverille, lukee kentän tiedot ja lisää ne uudelle sivulle. Tämän ansiosta kontrollien tila säilyy aina postbackin yhteydessä. [11, s. 134-139.]

```
<form method="post" action="State.aspx" id="form1">
...
      <input type="hidden" name="__VIEWSTATE" id="__VIEWSTATE"
      val-
      ue="IXcrUZ51B9YmtdoSL9csn2+VrYx5oW32kAw0oRXGsf3F0/W0l6/upie
      H7Nht1fhyr99U0IRRKmjvYk4FdH5E9ZRucaja0xPkwCyRoNBI3KkidqR5eA
      VX86DqOfEl584eSB0ff3IF4o3Y+ZqD7qZp3A==" />
```

Koodiesimerkki 6. View State –toiminnallisuuden luoma __VIEWSTATE-kenttä [11, s. 138].

5.6 ASP.NET sivun elinkaari

Ajatellaan sivun lähetystä web-palvelimelta selaimelle sivun elinkaarena. Sivun elinkaareen kuuluu useitä tärkeitä hetkiä. Kaari alkaa, kun selain tekee ensimmäisen serveripyynnön sivusta ja loppuu pian sen jälkeen, kun sivun koko HTML-sisältö on lähetetty selaimelle. ASP.NET tarjoaa 15 eri sivun tapahtumaa, jotka tapahtuvat serveripuolella automaattisesti jokaisen sivulatauksen yhteydessä. Metodeja, joita kutsutaan näiden tapahtumien yhteydessä, kutsutaan tapahtumakäsittelijöiksi. Tapahtumakäsittelijään kehittäjä sijoittaa toiminnallisuuden, joka suoritetaan, kun kyseinen tapahtuma tapahtuu. Tapahtumakäsittelijöiden avulla kehittäjä pääsee suorittamaan tehtäviä tietysssä kohdassa sivun elinkaarta.

Seuraavassa listassa käydään läpi sivun elinkaari kahdeksassa laajassa vaiheessa, joiden aikana tapahtuu vähintään yksi sivun tapahtuma.

1. **Sivun pyyntö.** ASPX-sivun pyyntö aloittaa sivun elinkaaren. Koko sivun elinkaarta ei suoriteta, jos web-palvelimen sallitaan palauttaa välimuistissa oleva kopio sivusta, muuten sivu etenee alkuvaiheeseen.

2. **Alku.** Tässä vaiheessa sivu saa pääsyn ominaisuuksiin, joita käytetään vuorovaikutukseen sivun ympäristön kanssa, kuten Request ja Response. Tämän lisäksi laukeaa PreInit-tapahtuma, joka tapahtuu ennen sivun alustusta.

3. **Sivun alustus.** Tämän vaiheen aikana sivun serverikontrollit tulevat käytettäviksi. Sivulla laukeaa Init-, InitComplete- ja PreLoad-tapahtumat.

4. **Sivun lataus.** Jos sivu on lähetetty takaisin (postback), tässä vaiheessa ladataan View State tiedot eli asetetaan kontrollit samaan tilaan, kuin ne olivat ennen uudelleenlähetystä. Esimerkiksi jos pudotusvalikosta valitaan arvo ja sen jälkeen lähetetään sivun takaisin, tässä vaiheessa arvo valitaan uudestaan alustettuun pudotusvalikkoon. Tämän jälkeen laukeaa Load-tapahtuma, jonka tapahtumakäsittelijässä kehittäjä yleensä tekee sivun alkutoimet.

5. **Validointi.** Tässä vaiheessa validointikontrollit, jotka tarkastavat käyttäjän antamat arvot, prosessoidaan. Jos validointi osoittaa syötteet vääränlaisiksi, seuraava vaihe jätetään väliin ja sivulla näytetään virheviesti.

6. **Postback-tapahtumien hallinta.** Tämän vaiheen aikana sivun kontrollit voivat laukaista omia tapahtumiaan. Esimerkiksi pudotusvalikko voi kutsua SelectedIndexChanged-tapahtuman tapahtumakäsittelijää kun käyttäjä on valinnut valikosta arvon tai nappi voi laukaista Clicked-tapahtuman kun sitä klikataan. Kontrollien tapahtumien jälkeen laukeaa LoadComplete-tapahtuma. Tämän vaiheen lopussa tapahtuu PreRender-tapahtuma, joka ilmaisee, että sivu on siirtymässä renderöintiin. Kun sivu on tallettanut kaikki tarvittavat View State tiedot, laukeaa SaveStateComplete-tapahtuma.

7. **Renderöinti.** Renderöinnissä kontrollit ja itse sivu lähettävät HTML-sisältönsä selaimelle.

8. **Purkaus.** Purkaus eli siivous –vaiheessa sivu ja kontrollit voivat vapauttaa käyttämänsä resurssit. Unload-tapahtuman tapahtumakäsittelijää kutsutaan, jotta kehittäjä voi hoitaa mahdolliset siivoukset manuaalisesti. [11, s. 219-220.]

5.7 Sivujen yhtenäisyys

Useimmissa web-sivustoissa ainoastaan osa sivusta muuttuu sivujen välillä liikuttaessa. Sivujen pysyvät osat koostuvat yleensä ylä- ja alatunnisteesta ja valikoista. ASP.NET tarjoaa yksinkertaisen tavat luoda ja hallita näitä pysyviä osioita toistamatta samaa koodia jokaisella sivuston sivulla. Tämän ratkaisun nimi on Master Page eli perustyylisivu. Perustyylisivulla voidaan luoda muotoilut ja pysyvät osiot yhdelle sivulle, jotka periytyvät kaikille sisältösivuille. Eli jos halutaan tehdä muutos esimerkiksi sivun valikkoon, tehdään muutos vain perustyylisivulle, jolloin muutos näkyy automaattisesti kaikilla siitä periytyvillä sivuilla.

Perustyylisivu muistuttaa joissain määrin normaalia ASPX-sivua, se sisältää normaalisti HTML-koodia ja myös serverikontrolleja ja sillä on myös oma koodisivunsa. Perustyylisivu ei kuitenkaan ole ei kuitenkaan ole todellinen ASPX-sivu eikä sitä voida ladata suoraan selaimella. Se toimii ainoastaan mallipohjana, joihin aidot web-sivut eli sisältösivut pohjaavat.

Kuten mainittu perustyylisivu sisältää HTML-koodia ja serverikontrolleja, joista sivuston pysyvät osat rakennetaan. Kohdat, jotka sisältösivut täyttävät, merkataan koodiesimerkki 7:n mukaisesti ContentPlaceHolder-kontrolleilla.

```
<asp:ContentPlaceHolder ID="ContentPlaceHolder1" runat="server">
</asp:ContentPlaceHolder>
```

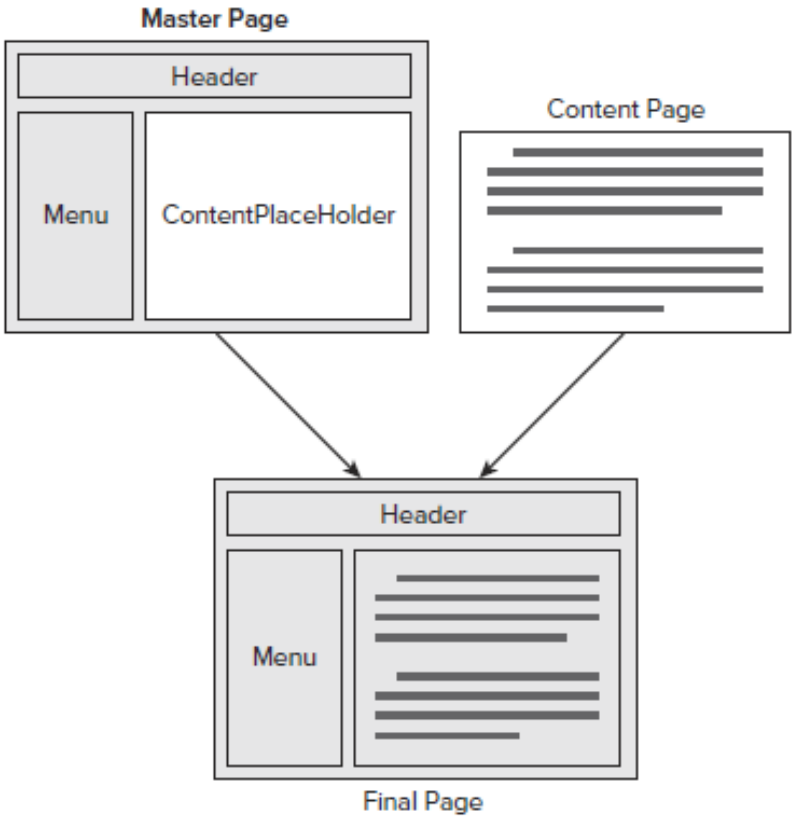
Koodiesimerkki 7. ContentPlaceHolder-serverikontrolli.

ContentPlaceHolder-kontrolleita voi luoda rajoittamattoman määrän, mutta useimmiten niitä ei tarvita kuin muutama. Sisältösivuilla määritellään sisällön kuuluminen tietyn ContentPlaceHolderin sisälle Content-kontrollilla. Kontrolli sitoutetaan ContentPlaceHolderiin koodiesimerkki 8:n mukaisesti ContentPlaceHolderID-attribuutilla ja haluttu sisältö kirjoitetaan kontrollin sisään.

```
<asp:Content ID="Content1" ContentPlace-
HolderID="ContentPlaceHolder1" runat="Server">
</asp:Content>
```

Koodiesimerkki 8. Content-serverikontrolli.

Näiden serverikontrollien avulla saadaan toteutettua helposti useita sivuja kuvan 7:n mukaisesti. Eli perustyylisivusta ja sisältösivusta muodostuvat lopullinen sivu, jonka selain lataa sisältösivun osoitetta pyydettäessä. [11, s. 208-210.]



Kuva 7. ASP.NET-sivun sisällön rakenne [11, s. 210].

5.8 ASP.NET AJAX

Asynchronous JavaScript And XML eli AJAX mahdollistaa web-sivun selainpuolen ja serveripuolen tiedonvaihdon asynkronisten kutsujen kautta. Asynkronisten kutsujen etu on, että ne eivät ajettaessa estä käyttöliittymän toimintaa. ASP.NET AJAX mahdollistaa:

- Vilkkumattomat sivut, joissa sivua ei tarvitse ladata kokonaan uudelleen postback-tapahtumassa. Voidaan ladata vain osa sivusta uudelleen, vaikuttamatta muihin osiin.
- Latauksen tilan näyttämisen käyttäjälle, esim. tilapalkkina tai lataushyrränä.
- Sivun päivittämisen ja serveripuolen koodin kutsumisen ajastettuna.

ASP.NET AJAX sivujen rakentaminen on erittäin yksinkertaista, sivulle tarvitaan vain ScriptManager ja UpdatePanel kontrollit. UpdateProgress ja Timer kontrollit tarjoavat pidemmälle vietyä toiminnallisuutta. [11, s. 350-352.]

UpdatePanel-serverikontrolli

UpdatePanel-kontrolli määrittää sivulta alueen, jota päivitetään vaikuttamatta sivun muihin osiin. Esimerkkinä jos painetaan UpdatePanelin sisällä olevaa nappia, joka aiheuttaa postback-tapahtuman, päivitetään vain kyseisen paneelin sisäinen alue. Vaikka vain osa sivusta päivitetään, koko sivu ja sen data lähetetään takaisin serverille postback-tapahtumassa. Sivu myös käy aina läpi tavanomaisen elinkaaren, jonka jälkeen serveri palauttaa selaimelle takaisin vain UpdatePanelin määrittelemän alueen HTML-koodin. [11, s. 352-356.]

ScriptManager-serverikontrolli

ScriptManager-kontrolli toimii siltana selainpuolen sivun ja serverin välillä. Se hallitsee skriptiresursseja eli selaimen käyttämiä JavaScript-tiedostoja ja hoitaa osittaisia sivun päivityksiä. ScriptManager-kontrolleja voi olla yhdellä sivulla vain yksi, se voidaan myös sijoittaa perustyylisivulle, jolloin jokainen sivu käyttää samaa kontrollia. [11, s. 356-357.]

UpdateProgress-serverikontrolli

UpdateProgress-kontrollia käytetään kertomaan käyttäjälle, että sivua päivitetään, se voidaan myös liittää tiettyyn UpdatePaneliin, jolloin se kertoo ainoastaan tämän paneelin päivittämisestä. Kontrollilla voidaan toteuttaa esimerkiksi koodiesimerkki 9:n mukainen lataushyrrä. ProgressTemplate-tunnisteiden sisälle jäävä alue näytetään sivulla vain, kun tietty UpdatePanel on ladannut vähintään sekunnin verran. Kun lataus on suoritettu loppuun, alue katoaa näkyvistä. DisplayAfter-attribuuttia käytetään määrittämään maksimi aika, jonka paneeli saa ladata ilman alueen näyttämistä. Tässä tapauksessa sivulla näkyy kuvan 8 mukainen lataushyrrä. [11, s. 358-361.]

```
<asp:UpdateProgress ID="UpdateProgress1" DisplayAfter="1000"
 AssociatedUpdatePanelID="UpdatePanel1" runat="server">
     <ProgressTemplate>
          <asp:Image      ID="Loading"      ImageUrl="loading.gif"
          runat="server" /> Please Wait…
```

```
    </ProgressTemplate>
</asp:UpdateProgress>
```

Koodiesimerkki 9. Lataushyrrän toteutus UpdateProgress-serverikontrollilla.

Kuva 8. Lataushyrrä selaimella tarkasteltuna.

Timer-serverikontrolli

Timer-kontrollilla voidaan helposti toteuttaa toistuvia toimintoja. Esimerkiksi sivua tai sen osaa voidaan päivittää tietyin aikavälein. Kontrollille voidaan myös luoda oma tapahtumakäsittelijänsä, jolloin ajastimen toiminta ei ole sidottu mihinkään sivun osaan. [11, s. 362.]

6 NETadmin-verkonhallintasovellus

Tässä projektissa toteutettava sovellus on rakennettu osaksi NETadmin-verkonhallintasovellusta. Yritys, jossa työskentelen, tekee NETadminin jatkokehitystä, joten useimmat asiakkaamme käyttävät sitä. Tämän takia myös häiriötiedotussovelluksen toteutus oli luonnollista tehdä osaksi NETadminia. Tämä luku esittelee NETadmin-verkonhallintasovelluksen toiminnot.

6.1 Yleistä

NETadmin-verkonhallintasovelluksella internetpalveluntarjoajan on mahdollista mm. hallita tietoverkkoja ja niiden tarjoamia palveluja, monitoroida palvelujen toimintaa ja ylläpitää asiakastukea. Sovelluksesta löytyvät toiminnot resurssien käytön suunnitteluun, vikailmoitusten hallintaan, laskutukseen, palvelutasosopimustilastointiin, verkkolaitteiden hallintaan ja verkon vianetsintään. NETadmin ei ole sidottu minkään laitevalmistajan tuotteisiin ja on myös täysin riippumaton verkkolaitteissa käytettävistä datan saantimenetelmistä. Tämä tarkoittaa, että NETadmin toimii äänen ja tiedon välittämiseen käytettyjen laajakaistaverkkojen kanssa ja tukee molempia langattomia ja

langallisia yhteyksiä. Tuettuihin laajakaistapalveluihin kuuluvat mm. nopeat internet yhteydet, Internet Protocol Television (IPTV), IP-puhelut, Point-to-Point yhteydet ja Virtual Private Network (VPN).

6.2 Toiminnot

End-to-End ratkaisu

NETadmin tukee ja automatisoi hallinnollista liikennettä teknikoiden, asiakaspalvelun, palveluntarjoajien ja loppukäyttäjien välillä. Jokaiselle osapuolille on oma räätälöity käyttöliittymänsä tai portaalinsa, jossa on vain heille tarpeellisia tietoja.

Automaattinen provisionti

Yksi NETadminin etuja on mahdollisuus aktivoida asiakkaiden palveluja automaattisesti. Automaation ansiosta asiakas voi selata ja valita haluamansa palvelut yhdeltä tai useammalta palveluntarjoajalta loppukäyttäjän portaalista, jonka jälkeen kaikki tarvittavat verkkoyhteydet, hallinnolliset muutokset ja inventaariopäivitykset tehdään automaattisesti minuuteissa.

Palveluiden valvonta

NETadmin valvoo laitteita, palveluja ja käyttöaktiivisuutta verkossa ja löytää nopeasti sen mahdolliset viat. Vikahälytyksiä voidaan, käyttöliittymässä näyttämisen lisäksi, lähettää sähköpostein ja tekstiviestein ja niiden määrää rajoitetaan verkonvalvojien suodatuksella ja ryhmittelyllä. Systeemi tarjoaa helposti lähestyttävän tavan valvoa palveluiden statistiikkaa, joka mahdollistaa verkon optimoinnin ja jatkokehityksen. [15.]

6.3 Historia

NETadmin sai alkunsa Wasadata Systems operaattorin sisäisesti kehitettynä ratkaisuna. Vuonna 2004 Netadmin Systems AB perustettiin jatkamaan systeemin kehitystä ja samana vuonna julkaistu NETadmin 7.0 versio oli sovelluksen ensimmäinen itsenäinen versio. Uusin 8.3-versio julkaistiin maaliskuussa 2013 ja sillä on noin 60 prosentin markkinaosuus Ruotsissa sekä asennuksia yli 15 maassa.

Asennuskanta koostuu yli sadasta asiakkaasta, jotka tuottavat palveluita yli miljoonalle loppukäyttäjälle. [16.]

7 Sovelluksen toteutus

Koska sovellus luotiin NETadmin 8.6 sivustolle raporttisivuksi, ainoat tarvittavat alkutoimenpiteet olivat sivun luominen NETadminin reports -kansioon ja määrittäminen raporttisivuksi NETadminiin. Tässä luvussa kuvataan sovelluksen käyttöliittymän eri näkymien toetutus.

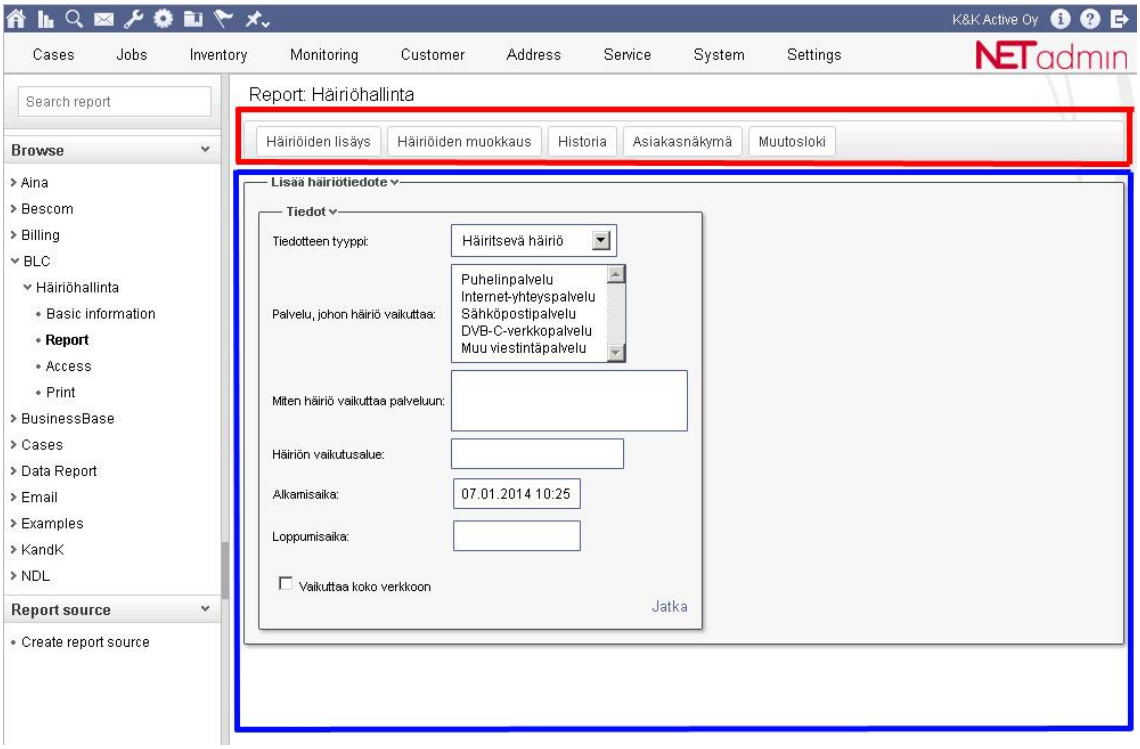
7.1 Sovellus NETadminin osana

ASP.NET-sivuston aspx-sivun koodin alussa määritellään aina sivun ominaisuudet. Raporttisivun tapauksessa määritellään koodiesimerkki 10:n mukaan ensin sivussa käytettävä ohjelmointikieli, kooditiedosto ja perustyylisivu. Raporttisivut käyttävät MasterPageMainReport.master-perustyylisivua, jotta NETadminin valikot pysyvät aina samana. ja raporttisivun sisältö sijoittuu oikeaan paikkaansa kuvassa 10 sinisellä merkittyyn alueeseen. Tämän jälkeen sivun käyttöön määritellään kaksi assemblyä, NetadminGUIControls antaa sivuston käyttöön NETadminin käyttöjärjestelmän ominaisuuksia ja Ext.Net erilaisia palvelinkontrolleja.

```
<%@ Page Language="vb" MasterPage-
File="~/common/MasterPageMainReport.master" Code-
File="fault_management.aspx.vb" Inherits="fault_management" %>
<%@ MasterType TypeName="MasterPageMainReport" %>
<%@ Register Assembly="NetadminGUIControls"
Namespace="Netadmin.GUI.Controls.Forms.Standard" TagPrefix="cc"
%>
<%@ Register Assembly="Ext.Net" Namespace="Ext.Net" TagPre-
fix="ext" %>
```

Koodiesimerkki 10. Raporttisivun NETadmin-yhteensopivuuden määrittely.

Kooditiedostossa luokka määritellään perimään NetadminReportPage-luokka, jotta sivun käyttöön saadaan kuvassa 9 punaisella merkattu valikkorivi.



Kuva 9. Häiriöhallinta-raporttisivu aloitustilassa NETadmin 8.6 -ympäristössä.

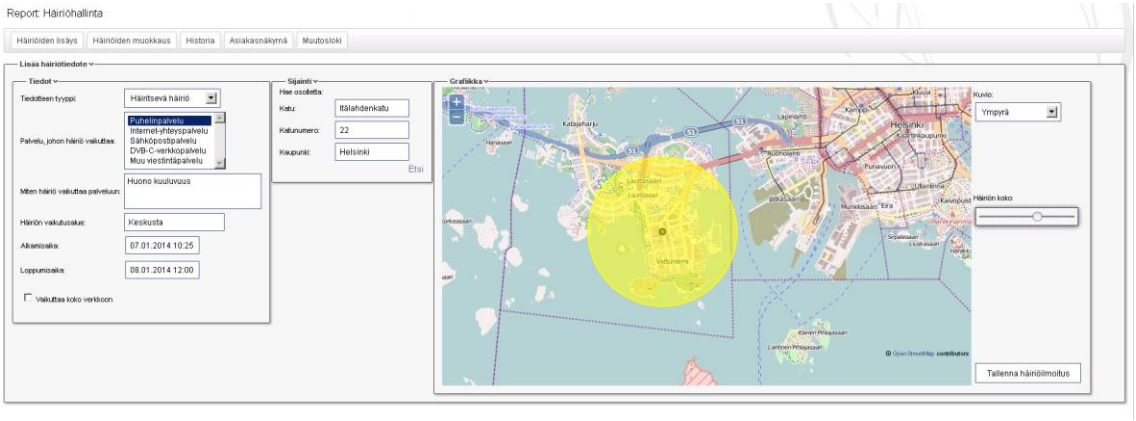
7.2 Sovelluksen käyttöliittymä

Sovelluksen päävalikkona toimii kuvassa 9 punaisella merkattu valikkorivi, josta käyttäjä pääsee vaihtamaan viiden sovelluksen päänäkymän väliltä. Näkymistä kaikki paitsi Asiakasnäkymä sijaitsevat samalla websivulla, jolloin tietty osa sivun sisällöstä piilotetaan ja osa näytetään. Asiakasnäkymä-nappi avaa selaimesta uuden ikkunan, jossa näytetään asiakkaille soveltuvat tiedot, muut osat sovelluksesta ovat vain ylläpitäjien käyttöön. Asiakasnäkymä voidaan upottaa esimerkiksi iframe HTML-tunnisteella internetpalveluntarjoajan kotisivuille.

7.3 Häiriötiedotteiden lisäys näkymä

Sovelluksen teknisesti suurin ja monipuolisin osuus on Häiriöiden lisäys näkymä. Se koostuu kuvan 10 mukaisesti kolmesta paneelista: Tiedot, Sijainti ja Grafiikka. Paneelit näytetään tietyssä järjestyksessä vasemmalta oikealle. Kun paneelin tiedot on täytetty, näytetään seuraava paneeli, viimeisestä paneelista löytyy häiriötiedotteen tallennusnappi. Tosin jos käyttäjä valitsee Tiedot-paneelista löytyvän ”Vaikuttaa koko

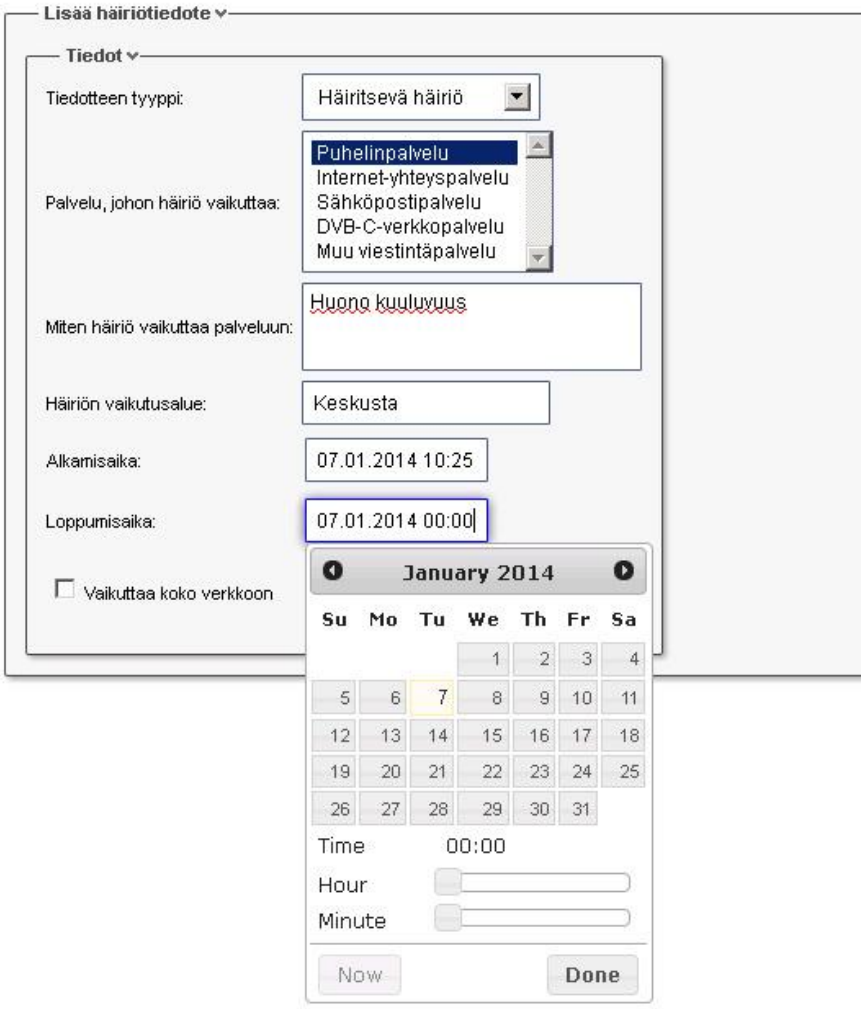
verkkoon” –kohdan jää tämä ainoaksi näytettäväksi paneeliksi, koska koko verkkoon vaikuttavat viat eivät tarvitse sijaintitietoja.



Kuva 10. Sovelluksen Häiriön lisäys näkymä.

7.3.1 Tiedot-paneeli

Tiedot-paneelissa määritellään häiriötiedotteen perustiedot: tiedotteen tyyppi, palvelu, johon häiriö vaikuttaa, miten häiriö vaikuttaa palveluun, häiriön vaikutusalue, häiriön alkamisaika ja arvioitu loppumisaika sekä vaikuttaako häiriö koko verkkoon. Häiriön tyyppi ja palvelu valitaan ennalta määritellyistä vaihtoehdoista, muut tiedot käyttäjä kirjoittaa itse. Aikatietojen valitsemiseksi paneelissa käytetään kuvassa 11 näkyvää DateTimePicker-jQuery komponenttia. jQuery on JavaScript-kirjasto, joka helpottaa HTML-sivun manipulointia. [17.]



Kuva 11. Tiedot-paneeli, jossa näkyvillä DateTimePicker-komponentti.

DateTimePicker asetetaan koodiesimerkki 11:sta mukaisesti JavaScript-koodina, ”.datepicker” määrittää, mihin kohtaa sivulla kyseinen koodi vaikuttaa. Tässä tapauksessa kaikkiin sivun tekstilaatikoihin, joiden class-arvona on ”datepicker”. Komponentista valitaan päivämäärä ja kellonaika, jonka jälkeen ne tulevat näkyviin tekstilaatikkoon koodiesimerkissä määritellyn muodon mukaisesti.

```
<script type="text/javascript">
   $(function () {
          $(".datepicker").datetimepicker({ dateFormat:
          'dd.mm.yy', timeFormat: 'HH:mm' });
   });

Sys.WebForms.PageRequestManager.getInstance().add_endRequest(
   function (sender, args) {
          $(".datepicker").datetimepicker({ dateFormat:
          'dd.mm.yy', timeFormat: 'HH:mm' });
```

```
    });
</script>
```

Koodiesimerkki 11. DateTimePicker-komponentin alustus.

7.3.2 Sijainti-paneeli

Sijainti-paneelissa käyttäjä etsii katuosoitteella maantieteellisiä koordinaatteja Maanmittauslaitoksen maastotietokannan osoitteiden kyselypalvelulla. Kuvan 12 mukaisesti käyttäjä syöttää kadun, katunumeron ja kaupungin, joille koordinaatteja etsitään.

Kuva 12. Sijainti-paneeli.

Tämän jälkeen sovellus lähettää koodiesimerkki 12:n mukaisesti Hypertext Transfer Protocol Secure (HTTPS) -pyynnön osoitepalvelun serverille. Pyyntö lähetetään muuten normaalia sivuhakua muistuttavana osoitteena, mutta se sisältää haun rajaavat tiedot XML-muotoisena. Haku vaatii käyttäjänimen ja salasanan koska kyseessä on maksullinen palvelu, joka veloittaa kyselyiden määrän mukaisesti. Palvelu palauttaa koordinaatit XML-muodossa, josta ne parsitaan ja tallennetaan Grafiikka-paneelin käyttöön. [18.]

```
Dim Xhttp As New MSXML2.XMLHTTP60

Dim AddressReq As String = "<Filter>" & _
     "<And>" & _
        "<PropertyIsEqualTo>" & _
            "<PropertyName>oso:kuntanimiFin</PropertyName>" & _
            "<Literal>" & City & "</Literal>" & _
        "</PropertyIsEqualTo>" & _
        "<PropertyIsEqualTo>" & _
            "<PropertyName>oso:katunimi</PropertyName>" & _
```

```
            "<Literal>" & Street & "</Literal>" & _
        "</PropertyIsEqualTo>" & _
        "<PropertyIsEqualTo>" & _
            "<PropertyName>oso:katunumero</PropertyName>" & _
            "<Literal>" & StreetNo & "</Literal>" & _
        "</PropertyIsEqualTo>" & _
    "</And>" & _
"</Filter>"

Dim UrlReq As String = Server.UrlEncode(AddressReq)

Xhttp.open("GET",
"https://ws.nls.fi/maasto/wfs?service=WFS&request=GetFeature&versi-
on=1.1.0&srsname=EPSG:4258&TYPENAME=oso:Osoitenimi&MAXFEATURES=1
&RESULTTYPE=results&NAMESPACE=xmlns(oso=http://xml.nls.fi/Osoitt
eet/Osoitepiste/2011/02)&filter=" & UrlReq, False, "Username",
"Password")
Xhttp.send()
```

Koodiesimerkki 12. Maastotietokannan osoitteiden kyselypalvelun HTTPS-pyynnön toteutus.

7.3.3 Grafiikka-paneeli

Grafiikka-paneelissa käyttäjä määrittelee kartalta alueen, jolla häiriö esiintyy. Kartta on toteutettu aiemmin esitellyillä OpenLayers- ja OpenStreetMap-tekniikoilla. Kuten koodiesimerkki 13:sta selviää, kartan luonnissa ensin luodaan Map-olio, johon kartan tasot lisätään. Tässä sovelluksessa käytetään kolmea tasoa: pohjatasona OpenStreetMap-taso ja sen lisäksi kaksi vektoritasoa. Vektoritasoista PolygonLayer sisältää kaikki käyttäjän piirtämät kuviot eli häiriöalueet, tasoa luodessa sille myös asetetaan kaksi eri muotoiluluokkaa, default määrittää vektorien muotoilun niitä lisätessä ja vertex niitä muokatessa. TargetLayeriin lisätään ainoastaan vektori merkkaamaan käyttäjän valitsemaa osoitetta.

```
var Map = new OpenLayers.Map("Kartta");

var OsmLayer = new OpenLayers.Layer.OSM("OpenStreetMap-taso");

PolygonLayer = new OpenLayers.Layer.Vector("Aluetaso", {
    styleMap: new OpenLayers.StyleMap({
         'default': DefaultStyle,
         'vertex': VertexStyle
    },
    { extendDefault: false }
    )
```

```
});

var TargetLayer = new OpenLayers.Layer.Vector("Kohdistustaso");

map.addLayers([OsmLayer, TargetLayer, PolygonLayer]);
```

Koodiesimerkki 13. OpenLayers-kartan tasojen luonti.

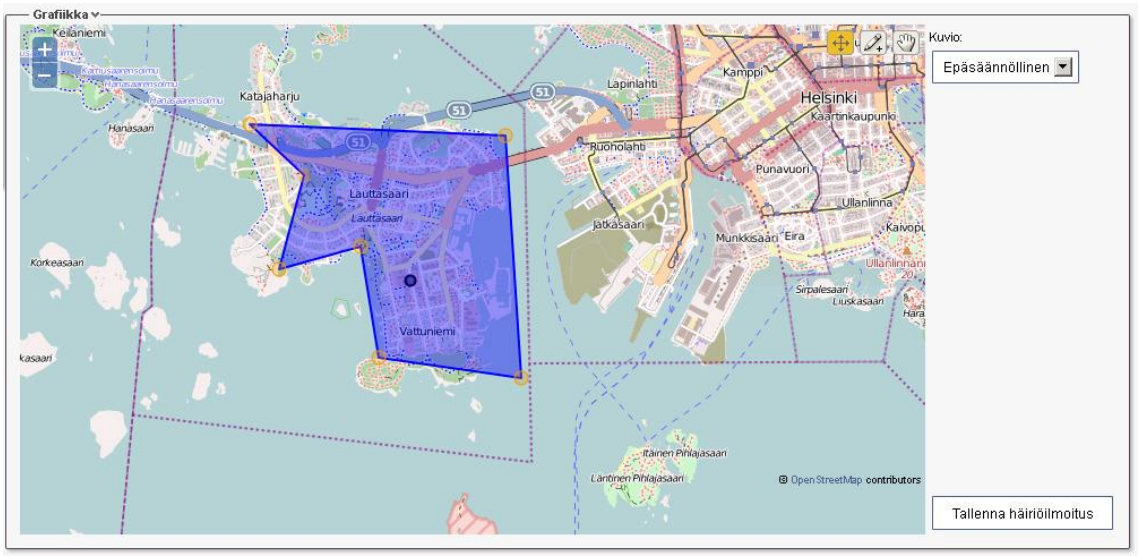
Grafiikka-paneelissa häiriöalue voidaan piirtää joko vapaasti lisäämällä pisteitä kartalle, jotka yhdistetään viivalla tai ympyränä, jonka keskipisteenä on käyttäjän antama osoite. Vapaasti piirtämiseen tarvitaan OpenLayersin tarjoamia kontrolleja. Koodiesimerkissä 14 luodaan nämä kontrollit ja lisätään ne karttaan. Kontrollit tarvitsevat ainoastaan nimen, tiedon mihin tasoon piirretään ja minkä muotoilun mukaan piirretään. Kuvassa 13 kartan oikeaan ylänurkkaan sijoittuvista kontrolleista on valittuna muokkaus-kontrolli, tällöin kartalle piirretystä kuviosta näytetään sen muodostavat pisteet. Kontrollin avulla pisteiden paikkaa ja määrää voidaan muokata.

```
var Panel = new OpenLayers.Control.Panel({
     displayClass: "EditingToolbar"
});

var Draw = new OpenLayers.Control.DrawFeature(
     PolygonLayer, { displayClass: "DrawFeaturePoint", title:
     "Piirrä" }
);

var Modify = new OpenLayers.Control.ModifyFeature(
     PolygonLayer, { displayClass: "ModifyFeature", title:
    "Muokkaa",       vertexRenderIntent: "vertex" }
);

Panel.addControls([
     new OpenLayers.Control.Navigation({ title: "Navigoi" }),
     Draw, Modify
]);

Map.addControl(Panel);
```

Koodiesimerkki 14. OpenLayers-kartan hallintakontrolleiden luonti.

Kun häiriötiedotteen tallennusnappia on painettu, tiedotteen tiedot tallennetaan MySQL-tietokantaan. Tietokantatoimintoihin NETadmin tarjoaa Netadmin Objects ja Netadmin Developers Library assemblyt. Näillä assemblyillä voidaan hakea tietoa joko suoraan MySQL-kyselyillä tai oliopohjaisesti, jolloin tiedot saadaan tietokannasta ilman

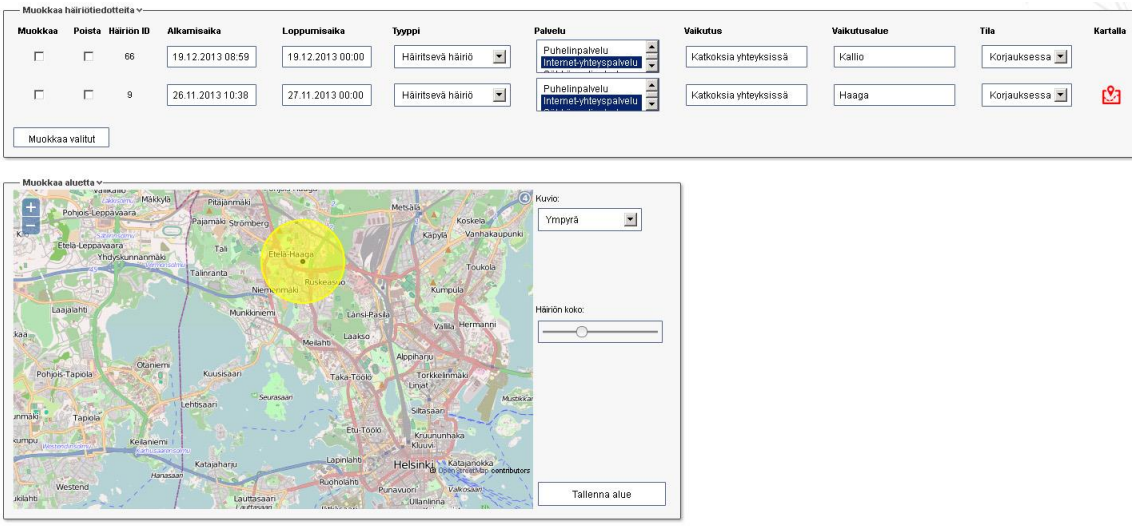
tietokannan rakenteen tuntemista. Molemmissa tapauksissa yhteys tietokantaan hoidetaan automaattisesti.



Kuva 13. Grafiikka-paneeli, jossa käytössä piirretyn alueen muokkaus.

7.4 Häiriötiedotteiden muokkaus näkymä

Häiriötiedotteiden muokkaus näkymässä käyttäjä voi muokata tiedotteen tietoja ja aluetta, merkata häiriön korjatuksi tai poistaa tiedotteen kokonaan. Kuvassa 14 häiriöalueen muokkaus on avattu toisen rivin tiedotteelle.



Kuva 14. Häiriötiedotteiden muokkaus näkymä.

Häiriöalueiden tiedot, eli vapaasti piirrettäessä kulmapisteiden koordinaatit ja ympyrän tapauksessa sen keskipiste ja säde, haetaan tietokannasta. Näiden tietojen mukaan piirretään muokattava häiriöalue kartalle. Koodiesimerkki 15 esittää ympyrävektorin piirtoon käytettävän metodin. Metodille annetaan säde ja koordinaatin pituus- ja leveysasteet, joista muodostetaan ensin piste-olio ja piste-oliosta monikulmio. Ympyrävektori ei siis ole aito ympyrä vaan koostuu monista kulmista, jotka muodostavat ympyräkuvion.

```
function DrawLoadedRoundVector(Radius, Longitude, Latitude) {

   var MapRadius = Radius / map.getExtent().getHeight();

   var Point = new OpenLayers.Geometry.Point(Latitude, Longi-
   tude).transform(new  OpenLayers.Projection("EPSG:4326"),
   Map.getProjectionObject());

   var NewPolygon = OpenLay-
   ers.Geometry.Polygon.createRegularPolygon(Point, MapRadius,
   40, 0);

   var PolygonVector =
   new penLayers.Feature.Vector(NewPolygon, null, null);

   PolygonLayer.addFeatures([PolygonVector]);
}
```

Koodiesimerkki 15. Funktio, jolla piirretään ympyrävektoreita OpenLayers-kartalle.

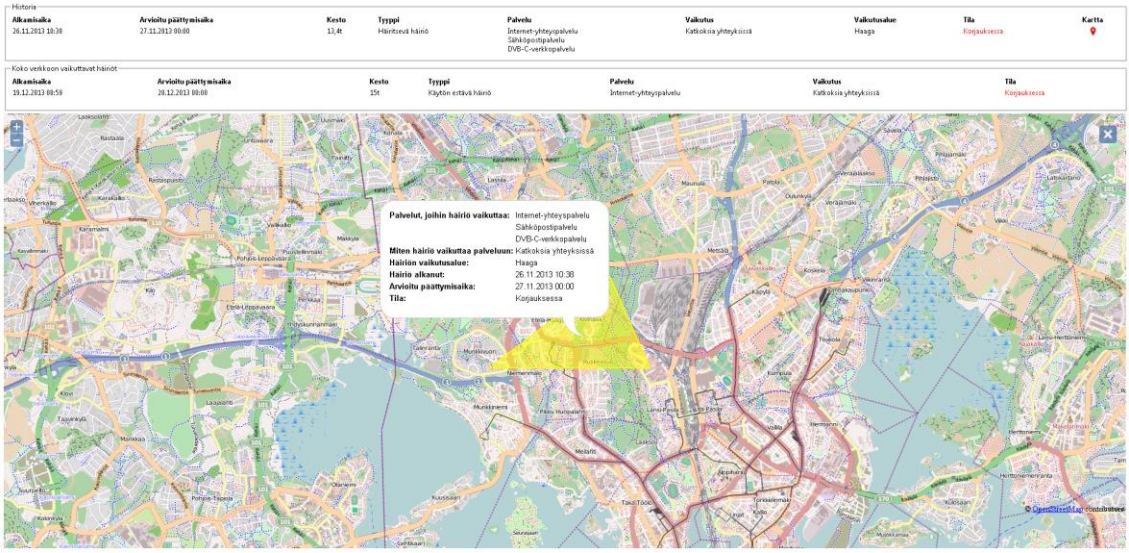
7.5 Historia-näkymä

Historia-näkymästä käyttäjä voi selata kaikkia työkalulla luotuja häiriötiedotteita, myös kuitattuja. Näkymän tiedot on myös mahdollista ladata Microsoft Excel tiedostona. Excel-käännöksen hoitaa NETadminin tarjoama GenerateExcelExport-metodi, joka lisää MySQL-tietokannasta haetun tiedon Excel-taulukkoon.

7.6 Asiakasnäkymä

Asiakkaille tarkoitettu Asiakasnäkymä koostuu kahdesta osasta. Kuten kuvassa 15 kaikkien alle kuukauden vanhojen häiriöiden tiedot näytetään ruudun yläreunassa. Painamalla häiriön Kartta-nappia kartta keskittää kyseisen häiriön alueeseen. Tämän

alapuolella listataan koko verkkoon vaikuttavat häiriöt. Näille häiriöille ei ole kartalla aluetta.



Kuva 15. Asiakasnäkymä, jossa näkyvillä häiriöalueen ponnahdusikkuna.

Häiriötietojen listauksen lisäksi näkymästä löytyy häiriökartta. Kartan toteutus on hyvin lähellä ylläpitopuolen karttaa, mutta mukana ei ole kuvioiden piirtoa ja aluetasoa luodessa määritellään kuvioille ponnahdusikkunat. Koodiesimerkki 16 esittelee ponnahdusikkunoiden toteutuksen. Tasoa luodessa määritetään kaksi tapahtumakäsittelijää: Featureselected ja Featureunselected. Featureselected-tapahtumakäsittelijä ajetaan kun jotakin tason kuviota klikataan. Tällöin näytetään käsittelijän määrittelemä ponnahdusikkuna. Esimerkissä Feature.attributes-olion tiedot ovat kuviota luodessa annettavia tietoja, jotka määrittelevät ponnahdusikkunan sijainnin ja sen sisältämän tiedon. Featureunselected-käsittelijä tuhoaa ponnahdusikkunan, kun jotain muuta kartan osaa klikataan.

```
var PolygonLayer = new OpenLayers.Layer.Vector("Aluetaso", {
  eventListeners: {
    'Featureselected': function (evt) {
        var Feature = evt.feature;
        var Popup = new OpenLay-
            ers.Popup.FramedCloud("Ponnahdusikkuna",
            new OpenLayers.LonLat(Feature.attributes.lati, Fea-
            ture.attributes.long).transform(new OpenLay-
            ers.Projection("EPSG:4326"),
            Map.getProjectionObject()),
            null,
            "<div style='font-size:.8em;font-family: Arial;'>" +
            Feature.attributes.html + "</div>",
```

```
                null,
                false,
                null
            );
            Popup.autoSize = true;
            Feature.popup = Popup;
            Map.addPopup(Popup);
        },
        'Featureunselected': function (evt) {
            var Feature = evt.feature;
            Map.removePopup(Feature.popup);
            Feature.popup.destroy();
            Feature.popup = null;
        }
    }
});
```

Koodiesimerkki 16. Vektoritason toteutus, jossa vektoreille luodaan ponnahdusikkunat.

7.7 Muutosloki-näkymä

Muutoslokiin tallennetaan kaikki sovelluksessa tehdyt muutokset tiedotteisiin. Lokiin merkatut käyttäjät ovat NETadmin-käyttäjätunnuksia, joiden tiedot saadaan raporttisivun perustyylisivulta.

8 Pohdinta

Projektin aikana olen erityisesti mieltynyt ASP.NET-ohjelmistokehykseen. Se nopeuttaa ja yksinkertaistaa websivustojen kehitystä huomattavasti. Sen tarjoamat monipuoliset serverikontrollit hoitavat toimintoja, joiden toteuttaminen veisi muuten projektilta paljon resursseja.

OpenStreetMapin ja OpenLayersin toimintaan olen tyytyväinen. Varsinkin OpenLayers tarjoaa todella kattavan kirjaston tasojen toteutukseen. Sen piirtokontrollien helppous teki vaikutuksen. Valitettavasti OpenLayersistä paistaa sen avoimen lähdekoodin kehitys. Kirjastosta löytyi useampia ohjelmointivirheitä, joita ei ollut vielä korjattu, ja jouduin käyttämään aikaa niiden korjaamiseen. OpenStreetMap oli tämän projektin tarpeisiin sopiva, sen karttamateriaaleista löytyi Suomesta kattavasti tieverkostot, rakennukset ja alueet. Verrattuna esimerkiksi Googlen karttapalveluun

OpenStreetMapin kartat Suomesta ovat lähes identtisiä eli Googlen karttapalveluiden käyttöoikeuden ostamisesta ei olisi ollut huomattavaa hyötyä.

Projektin ongelmallisin tekniikka oli Maanmittauslaitoksen osoitteiden kyselypalvelu. Vaikka systeemissä on erittäin kattavasti osoitetietoja, niiden haku on huonosti toteutettu. Haku ottaa etsintäparametreiksi kadun ja kunnan. Nykytavan mukaan osoitteet ilmoitetaan kadun ja kaupungin mukaan, eikä käyttäjällä ei aina ole tietoa mihin kuntaan kyseinen osoite kuuluu. Jos haku suoritetaan kaupungin nimellä, joka ei ole sama kuin kunnan nimi, haku epäonnistuu. Google tarjoaa koordinaatin hakuun palvelun, jonka hakupalvelu on selkeästi parempi ja se löytää oikeat koordinaatit myös kaupunkien mukaan. Valitettavasti palvelua saa käyttää ainoastaan kun käytössä on myös Googlen karttapalvelu, jolloin palveluiden yhteishinta nousee selkeästi korkeammaksi kuin Maanmittauslaitoksen palvelun hinta.

Projektin aloittaminen myöhästyi hieman asiakkaasta johtuneista seikoista, mutta kun sovellusta päästiin toteuttamaan, pysyttiin hyvin aikataulussa. Sovellukseen onnistuttiin toteuttamaan kaikki halutut toiminnallisuudet ja sen käyttöliittymästä saatiin sulava ja selkeä. Toteutuksen jälkeen asiakas testasi sovelluksen. Testauksesta saatiin positiivista palautetta, ja asiakas hyväksyi lopputuotteen.

Lähteet

1 Hazzard, Erik. 2011. OpenLayers 2.10 Beginner's Guide. Birmingham, Iso-Britannia: Packt Publishing Ltd.

2 Alessio Di Lorenzo Giovanni Allegri. 2013. What is OpenLayers? Verkkodokumentti. <http://www.packtpub.com/article/what-openlayers>. Luettu 20.12.2013.

3 Google Maps JavaScript API v3. 2013. Verkkodokumentti. <https://developers.google.com/maps/documentation/javascript/maptypes>. Luettu 21.12.2013

4 About MapGuide. 2006. Verkkodokumentti. <http://mapguide.osgeo.org/about.html>. Luettu 21.12.2013.

5 OpenLayers API. Verkkosivusto. <http://dev.openlayers.org/releases/OpenLayers-2.13.1/doc/apidocs/files/OpenLayers-js.html>.

6 Bing Maps REST Services. Verkkodokumentti. <http://msdn.microsoft.com/en-us/library/ff701713.aspx>. Luettu 27.12.2013.

7 Bennett, Jonathan. 2010. OpenStreetMap. Birmingham, Iso-Britannia: Packt Publishing Ltd.

8 ASP.NET. Verkkodokumentti. <http://en.wikipedia.org/wiki/ASP.NET>. Luettu 28.12.2013.

9 ASP.NET Tutorial. Verkkodokumentti. <http://www.w3schools.com/aspnet/>. Luettu 28.12.203.

10 Bochicchio, Daniele. Mostarda,Stefano. De Sanctis, Marco. 2011. ASP.NET 4.0 In Practise. USA: Manning.

11 Spaanjaars, Imar. 2013. Beginning ASP.NET 4.5: in C# and VB. Indianapolis, USA: John Wiley & Sons, Inc.

12 Viestintäverkkojen ja -palvelujen häiriöt. 2014. Verkkodokumentti. <https://www.viestintavirasto.fi/internetpuhelin/toimivuus/verkonhairiot.html>. Luettu 2.2.2014.

13 Iacovella, Stefano. Youngblood, Brian. 2013. GeoServer Beginner's Guide. Birmingham, Iso-Britannia: Packt Publishing Ltd.

14 ASP.NET Page Life Cycle Overview. 2011. Verkkodokumentti. <http://msdn.microsoft.com/en-us/library/ms178472.aspx>. Luettu 9.2.2014.

15 NETadmin, an OSS/BSS platform for broadband networks. Verkkodokumentti. <http://www.netadminsystems.com/products/product-menu-netadmin/overview>. Luettu 15.2.2014.

16 Netadmin Systems, the result of real-world experiences. Verkkodokumentti. <http://www.netadminsystems.com/company>. Luettu 15.2.2014.

17 jQuery. Verkkodokumentti <http://jquery.com/>. Luettu 8.3.2014.

18 Maastotietokannan osoitteiden kyselypalvelu (WFS). Verkkodokumentti. <http://www.maanmittauslaitos.fi/aineistot-palvelut/rajapintapalvelut/maastotietokannan-osoitteiden-kyselypalvelu-wfs>. Luettu 12.12.2013.